

APPARATUS AND METHOD FOR PASSIVELY MONITORING LIVENESS OF JOBS IN A CLUSTERED COMPUTING ENVIRONMENT

BACKGROUND OF THE INVENTION

1. Technical Field

5 This invention generally relates to data processing, and more specifically relates to the sharing of tasks between computers on a network.

2. Background Art

10 Since the dawn of the computer age, computer systems have become indispensable in many fields of human endeavor including engineering design, machine and process control, and information storage and access. In the early days of computers, companies such as banks, industry, and the government would purchase a single computer which satisfied their needs, but by the early 1950's many companies had multiple computers and the need to move data from one computer to another became apparent. At this time computer networks began being developed to allow computers to work together.

15 Networked computers are capable of performing tasks that no single computer could perform. In addition, networks allow low cost personal computer systems to connect to larger systems to perform tasks that such low cost systems could not perform alone. Most companies in the United States today have one or more computer networks.

20 The topology and size of the networks may vary according to the computer systems being networked and the design of the system administrator. It is very common, in fact, for companies to have multiple computer networks. Many large companies have a

sophisticated blend of local area networks (LANs) and wide area networks (WANs) that effectively connect most computers in the company to each other.

With so many computers hooked together on a network, it soon became apparent that networked computers could be used to complete tasks by delegating different portions of the task to different computers on the network, which can then process their respective portions in parallel. The concept of a computer “cluster” has been used to define groups of computer systems on a network that can work on predefined tasks.

If an error occurs while processing some task that is defined for a group of computers in a cluster, there needs to be some way to detect that the error has occurred. In addition, there needs to be some way to distinguish an error from a task that takes a substantial period of time to run to completion. One known way to detect errors and distinguish errors from long processing times uses the concept of the “liveness” of a job.

A job is the work that a computer does for a user. The “liveness” of a job refers to whether a job is correctly executing its program. Known methods for checking liveness use an active liveness monitoring process that runs on each node in a group. Active liveness monitoring means a job is explicitly checked for liveness. The active liveness monitoring process sends out periodic inquiries asking a group member if it is still alive, and awaits a response from that job. This is done for all jobs on a computer that are members of a group. Typically, a predetermined period of time, such as 1-3 seconds, is selected that is longer than the longest anticipated processing time for any group member job. If a group member job does not respond within the predetermined time period, the job is presumed dead, and the remaining jobs can then take appropriate action.

Active liveness monitoring can take considerable system resources. Each liveness monitoring process must check liveness of all jobs on its node, and must also check to see

if the other nodes are live as well. If the number of jobs and the number of nodes are high, the cluster may expend considerable and excessive resources performing the liveness checking of its members. Without a mechanism for passively monitoring liveness of group member jobs, the known active liveness checking will continue to be an
5 excessive drain on system resources.

DISCLOSURE OF INVENTION

An apparatus and method passively determine when a job in a clustered computing environment is dead. Each node in the cluster has a cluster engine for communicating between jobs within the same group on other nodes. A protocol is
10 defined that includes one or more acknowledge (ACK) rounds, and that only performs local processing between ACK rounds. The protocol is executed by jobs that are members of a defined group. Each job in the group has one or more work threads that execute the protocol. In addition, each job has a main thread that communicates between the job and jobs on other nodes (through the cluster engine), routes appropriate messages
15 from the cluster engine to a work thread, and signals to the cluster engine when a fault occurs when the work thread executes the protocol. By assuring that a dead job is reported to other members of the group, liveness information for group members can be monitored without the overhead associated with active liveness checking.

The foregoing and other features and advantages of the invention will be apparent
20 from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

The preferred embodiments of the present invention will hereinafter be described in conjunction with the appended drawings, where like designations denote like elements, and:

5 FIG. 1 is a block diagram of computer systems that may intercommunicate on a network;

 FIG. 2 is a block diagram of a prior art cluster node that includes an active liveness monitoring process;

 FIG. 3 is a flow diagram of a prior art method for actively checking the liveness
10 of group members jobs in a cluster;

 FIG. 4 is a block diagram of a node in accordance with the preferred embodiments that includes a main thread that indicates when its job is dead without any active liveness checks;

 FIG. 5 is a block diagram of the main thread in FIG. 4;

15 FIG. 6 is a block diagram of one of the work threads in FIG. 4;

 FIG. 7 is a block diagram of the protocol that is executed by the work thread of
FIG. 6;

 FIG. 8 is a block diagram of a computer system in accordance with the present invention that serves as a node in a cluster; and

20 FIG. 9 is a block diagram of a method in accordance with the preferred embodiments for processing faults encountered by a work thread.

BEST MODE FOR CARRYING OUT THE INVENTION

The present invention is accomplished through sharing portions of tasks on computers that are connected on a network. For those who are not familiar with networking concepts, the brief overview below provides background information that will help the reader to understand the present invention.

1. Overview

Networked Computer Systems

Connecting computers together on a network requires some form of networking software. Over the years, the power and sophistication of networking software has greatly increased. Networking software typically defines a protocol for exchanging information between computers on a network. Many different network protocols are known in the art. Examples of commercially-available networking software is Novell Netware and Windows NT, which each implement different protocols for exchanging information between computers.

One significant computer network that has recently become very popular is the Internet. The Internet grew out of a proliferation of computers and networks, and has evolved into a sophisticated worldwide network of computer systems. Using the Internet, a user may access computers all over the world from a single workstation. TCP/IP (Transmission Control Protocol/Internet Protocol) is an example of a network protocol that is in wide use today for communicating between computers on the Internet. In addition, the use of TCP/IP is also rapidly expanding to more local area networks (LANs) and Intranets within companies.

Computer Clusters

The prior art recognized the benefit of having groups of computer systems work on different pieces of a problem. The concept of “clusters” of computers evolved to include a predefined group of networked computers that can share portions of a larger task. One specific implementation of a cluster uses ordered messages for communicating between the computers in a cluster. In an ordered message system, each message is communicated to all nodes, and the order of messages is enforced so that all nodes see the messages in the same order.

Referring to FIG. 1, a simple cluster 100 of five computer systems (or “nodes”) 110 is shown. The connections between these nodes represents a logical connection, and the physical connections can vary within the scope of the preferred embodiments so long as the nodes in the cluster can logically communicate with each other. Within a cluster, one or more “groups” may be defined, which correspond to logical groupings of nodes that cooperate to accomplish some task. Each node in a group is said to be a “member” of that group. As shown in FIG. 2, each node 210 in a prior art cluster includes an active liveness monitoring process 220, a cluster engine 230 (referred to herein as CLUE), and one or more jobs 240. Each job 240 includes one or more work threads 250 that execute the job 240, which amounts to a portion of the larger task that is being delegated to the members of the group. The active liveness monitoring process 220 includes a timer 222 that is used to determine the amount of time that passes from the point in time that a transmission of a liveness message is sent to a group member. If a group member does not respond within a predetermined period of time as measured by the timer 222, the non-responding member is presumed to be dead, and the remaining members of the group may then take appropriate action.

CLUE 230 is a software process that enforces ordered messages between nodes in a cluster. All messages by any member of the group are communicated to the node's local CLUE 230, which then communicates the message to all other members of the group. When a job 240 wants to be part of a group, it registers with CLUE 230 as a member of that group. This registration causes CLUE to generate a membership change message to other members of the group to inform the other members of the new addition to the group. In similar fashion, when a job 240 no longer wants to become a member of the group, it unregisters with CLUE 230, which also causes a corresponding membership change message to inform the remaining members of the group that a member has been deleted from the group. When CLUE 230 receives a message from its member that is intended for the group, CLUE 230 sends the message to all registered members.

Referring to FIG. 3, a method 300 represents steps that are suitably performed by the active liveness monitoring process 220 in FIG. 2. The active liveness monitoring process 220 sends out an active liveness message to each member on this node (step 310). Each member responds back to the active liveness message an acknowledgment (ACK) or negative acknowledgment (NACK). An ACK means the member is functioning correctly, and a NACK means the member is not functioning correctly. The member checks itself for proper operation. These checks may include sending messages to other members in its group, checking the status of any currently executing protocol, or issuing a test/liveness protocol. After waiting a predetermined period of time (step 320), if all members of the group respond with an ACK (step 330=YES), method 300 loops back to step 310 and continues monitoring. If one or more members do not respond with an ACK (step 330=NO), this lack of response is treated as a failure, and the failure is processed appropriately (step 340). One known way for processing a failure is for CLUE to generate a membership change message to all group members when a member fails to respond during a liveness round. The active liveness monitoring process informs CLUE of a failing member and CLUE unregisters the failing member, and the resulting

membership change message that CLUE sends informs all other members of the failing member.

As stated in the Background section, a major problem with the prior art active liveness monitoring described above with reference to FIGS. 2 and 3 is the constant drain of system resources to continually check the liveness of members of a group. A second problem with the prior art is the potential complexity of the active liveness check by a monitor and a member. It is difficult for a monitor to know if a timer expiration is due to a member job being slow to react, or due to a true error that has occurred. The failing member job needs to have itself terminated before another member can take over the failing member's duties, so the monitor needs to know not only when the failing job actually ends, but also may have to terminate it. Another problem with active liveness monitoring is that a liveness check can occur at any time, so a member may need to execute concurrently two protocols: the liveness check, and a currently executing protocol. Designing concurrent protocols is generally considered to be significantly more difficult than designing non-concurrent protocols. Yet another problem is the reliability of the active liveness monitor. Should the active liveness monitor fail, the system will never know of the failure. The present invention as described in the description of the preferred embodiments below provides an apparatus and method for passively monitoring liveness of group members, that only signals when a member is dead.

2. Detailed Description

According to a preferred embodiment of the present invention, an apparatus and method for passively monitoring liveness of group members overcomes the disadvantages of prior art active liveness monitoring schemes by providing an architected way for a group member to reliably report if its job is dead without having an active liveness monitoring process running on each node. There is very little overhead until an

error is occurred, at which time the error can be processed appropriately. Thus, the significant overhead in prior art active liveness monitoring schemes has been all but eliminated, thereby greatly enhancing system performance.

Referring now to FIG. 4, a node 410 represents a node in a cluster, such as that shown in FIG. 1. Node 410 in accordance with the preferred embodiments includes a cluster engine (CLUE) 230, and one or more jobs 440. Each job 440 has one or more corresponding work threads 450. In addition, each job 440 also includes a single main thread 442 that is not found in the prior art.

FIG. 5 illustrates one suitable implementation for main thread 442, which includes a message monitor and router 510, a kill mechanism 520, and a CLUE registration/unregistration mechanism 530. The main thread 442 does not do any work on the job 440, but is a supervisory mechanism that passes messages from CLUE 230 to the work thread(s), and that detects when a work thread is not longer alive. The message monitor and router 510 monitors all messages received from CLUE 230. If a message is a special type of message (known as a "kill message") that tells the job to kill a specified work thread, main thread 442 uses the kill mechanism 520 to kill the specified work thread 450. Main thread 442 can also use kill mechanism 520 to kill a protocol by sending an abort message to a work thread. The work thread, in turn, sends a NACK message, which causes the protocol to abort. In addition, if the main thread 442 kills a work thread 450, whether by request of the work thread or by some unrecoverable error that occurs in the work thread, main thread 442 then unregisters with CLUE using the CLUE registration/unregistration mechanism 530. By unregistering with CLUE, all other members of the group know that the failing member is no longer a member of the group, and the remaining members can then process the error or take other appropriate action.

One suitable implementation of a work thread 450 in FIG. 4 is shown in FIG. 6. Work thread 450 is a thread of execution that actually performs the work of its corresponding job, as defined in a protocol 610. Protocol 610 is comprised of multiple phases that work thread 450 can perform. The present invention is made possible by
5 defining certain characteristics of the main thread 442 and by defining certain characteristics of protocol 610.

For the preferred embodiments, group member liveness is defined to have two conditions: 1) responsiveness; and 2) progress. Responsiveness means that a member is able to read group messages. Progress means working meaningfully on a protocol (*e.g.*,
10 not in an infinite wait or in an endless loop). If a group member is responsive and is making progress, then it must be live.

The responsiveness of a group member is assured in the preferred embodiments by having a main thread 442 in each job 440. Main thread 442 performs limited functions that assure responsiveness. The main thread 442 only reads messages, forwards
15 protocol messages to its work thread(s) 450, prioritizes messages as needed, and executes special messages, such as messages to terminate a work thread or to terminate the member job. No main thread can do any work that could lead to it not being available to read messages. This means that the main thread 442 cannot generate any ACK rounds, because waiting on an ACK round could result in the main thread being unavailable to
20 read a message. In addition, the main thread 442 cannot wait to acquire a local resource (but it can still do work), because waiting may also result in the main thread being unavailable to read a message. By defining the main thread in this manner, we know that the main thread will always be available to read a message, so the other group members need not be concerned that a sent message was not received by another member. This
25 means that each member can send messages to the group without explicitly checking to see if the messages were received.

Progress of a group member is assured by appropriately defining the structure of the protocol 610. Referring to FIG. 7, a protocol in accordance with the preferred embodiments is divided into a number of different phases divided by ACK rounds. Each phase is defined in a way that assures that the member only does local work during a
5 phase. When information is needed from another member, the information is sent through CLUE and is followed by an ACK round. The result is that progress is ensured between ACK rounds, and any failure during an ACK round will be communicated by the failing member either issuing a NACK response or unregistering with CLUE. CLUE guarantees that if a member fails to respond during an ACK round (and thus unregisters
10 with CLUE), CLUE sends a special message known as a membership change to all members left in the group. The membership change is treated as a negative acknowledge (NACK) signal from the member that did not respond. In one embodiment, the remaining members in response to a NACK signal undo the changes that were made during execution of the protocol. In another embodiment, the remaining members may
15 determine that the failure of the dead member is not significant, and may then continue processing the protocol.

Because we know that the work thread 442 on each group member is always responsive, we know that each group member will receive and recognize the membership change. Because the work thread(s) only do local work between ACK rounds, the work
20 thread will always progress to an ACK round (assuming no local deadlock), so each member is assured to see the membership change. Defining protocols such that only local work is done between ACK rounds means that a group member will always progress to an ACK round. Providing a main thread for each group member means that a group member will always be responsive. By assuring both progress and responsiveness
25 in this manner, the present invention results in group members that will simply unregister with CLUE if an error occurs, resulting in a membership change message from CLUE to remaining group members. No active monitoring is required.

Another advantage of the present invention is that checking the state of a group member is relatively straightforward because the member is guaranteed to progress to an ACK round. All other members in a group will know how far each other member has progressed in the protocol by virtue of knowing which ACK round(s) have occurred. In addition, if more information about the failing member is desired, a query message may be sent to the work thread. Because the work thread is always responsive, it will receive the query and can respond to it. The result is that the query occurs at a well-defined point in a protocol, rather than at any time.

Referring to FIG. 8, a computer system 800 is an enhanced IBM AS/400 computer system, and represents one suitable type of node 410 (FIG. 4) that can be networked together in accordance with the preferred embodiments. Those skilled in the art will appreciate that the mechanisms and apparatus of the present invention apply equally to any computer system that can be networked together with other computer systems. As shown in FIG. 8, computer system 800 comprises a processor 810 connected to a main memory 820, a mass storage interface 830, a terminal interface 840, and a network interface 850. These system components are interconnected through the use of a system bus 860. Mass storage interface 830 is used to connect mass storage devices (such as a direct access storage device 855) to computer system 800. One specific type of direct access storage device is a floppy disk drive, which may store data to and read data from a floppy diskette 895.

Main memory 820 contains data 822, an operating system 824, a cluster engine (CLUE) 230, and one or more jobs 440 that each contain a main thread 442 and one or more work threads 450. Data 822 represents any data that serves as input to or output from any program in computer system 800. Operating system 824 is a multitasking operating system known in the industry as OS/400; however, those skilled in the art will appreciate that the spirit and scope of the present invention is not limited to any one

operating system. CLUE 230 is a cluster engine that communicates with other computer systems in a defined cluster. In the preferred embodiments, CLUE 230 enforces ordered messages, which means that each member in the cluster will see messages in the same order. In the preferred embodiments, CLUE 230 is a known cluster engine with functions
5 as described above with respect to FIGS. 2-4. However, it is equally within the scope of the present invention to provide a cluster engine 230 that has new or different attributes when compared to known cluster engines.

A job 440 can be a member of a group on a cluster that executes a defined protocol. Each job contains one main thread 442 and one or more work threads 450. The
10 main thread 442 includes the features described above with reference to FIG. 5, and is defined to have no ACK rounds so it can never get stuck waiting on another member of the group. In addition, main thread 442 is defined in a way that assures it will never get stuck doing local work. This means, for example, that a main thread 442 cannot wait to acquire a local resource. The work thread(s) are described above with reference to FIG.
15 6. Each work thread executes a protocol 610 or a portion of a protocol 610, and communicates with the other group members (through the main thread 442 and CLUE 230) at ACK rounds defined in the protocol.

Computer system 800 utilizes well known virtual addressing mechanisms that allow the programs of computer system 800 to behave as if they only have access to a
20 large, single storage entity instead of access to multiple, smaller storage entities such as main memory 820 and DASD device 855. Therefore, while data 822, operating system 824, CLUE 230, and jobs 440 are shown to reside in main memory 820, those skilled in the art will recognize that these items are not necessarily all completely contained in main memory 820 at the same time. It should also be noted that the term "memory" is used
25 herein to generically refer to the entire virtual memory of computer system 800.

Processor 810 may be constructed from one or more microprocessors and/or integrated circuits. Processor 810 executes program instructions stored in main memory 820. Main memory 820 stores programs and data that processor 810 may access. When computer system 800 starts up, processor 810 initially executes the program instructions that make up operating system 824. Operating system 824 is a sophisticated program that manages the resources of computer system 800. Some of these resources are processor 810, main memory 820, mass storage interface 830, terminal interface 840, network interface 850, and system bus 860.

Although computer system 800 is shown to contain only a single processor and a single system bus, those skilled in the art will appreciate that the present invention may be practiced using a computer system that has multiple processors and/or multiple buses. In addition, the interfaces (called input/output processors in AS/400 terminology) that are used in the preferred embodiment each include separate, fully programmed microprocessors that are used to off-load compute-intensive processing from processor 810. However, those skilled in the art will appreciate that the present invention applies equally to computer systems that simply use I/O adapters to perform similar functions.

Terminal interface 840 is used to directly connect one or more terminals 865 to computer system 800. These terminals 865, which may be non-intelligent (*i.e.*, dumb) terminals or fully programmable workstations, are used to allow system administrators and users to communicate with computer system 800. Note, however, that while terminal interface 840 is provided to support communication with one or more terminals 865, computer system 800 does not necessarily require a terminal 865, because all needed interaction with users and other processes may occur via network interface 850.

Network interface 850 is used to connect other computer systems and/or workstations (*e.g.*, 875 in FIG. 8) to computer system 800 across a network 870.

Network 870 represents the logical connections between computer system 800 and other computer systems on the network 870. The present invention applies equally no matter how computer system 800 may be connected to other computer systems and/or workstations, regardless of whether the network connection 870 is made using present-day analog and/or digital techniques or via some networking mechanism of the future. In addition, many different network protocols can be used to implement a network. These protocols are specialized computer programs that allow computers to communicate across network 870. TCP/IP (Transmission Control Protocol/Internet Protocol) is an example of a suitable network protocol.

At this point, it is important to note that while the present invention has been and will continue to be described in the context of a fully functional computer system, those skilled in the art will appreciate that the present invention is capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution. Examples of suitable signal bearing media include: recordable type media such as floppy disks (*e.g.*, 895 of FIG. 8) and CD ROM, and transmission type media such as digital and analog communications links.

Referring to FIG. 9, a method 900 illustrates the steps a main thread 442 performs in processing an error in a group member during the execution of a protocol. Because the main thread is defined to have no ACK rounds, and because the protocol is defined to do only local work between ACK rounds, some of the error conditions that exist in prior art systems have been eliminated in the system and method of the preferred embodiments. With a system in accordance with the preferred embodiments, two errors 910 and 950 are considered. Step 910 is an error that occurs when a non-recoverable fault occurs in a work thread. When this happens, the main thread kills the work thread (step 920). The main thread then unregisters with CLUE (step 930), which causes CLUE to transmit the

membership change to the remaining group members (step 940). This membership change serves as notification to the other group members that the group member that unregistered with CLUE is no longer alive.

Another error that can occur is when a work thread 450 does not receive an ACK from all members during an ACK round (step 950). When this happens, the user is queried to determine which action the user wishes to take (step 960). If the user specifies that the work thread should be terminated (step 962=YES), the work thread is terminated by executing steps 920, 930 and 940 described above. If the user does not want to kill the work thread (step 962=NO), the user may instead specify to abort the protocol (step 964=YES), which causes the main thread 442 to send a NACK message. This NACK message informs the group that the member that sent the NACK is no longer alive. Because a membership change message received from CLUE is treated as a NACK message from the missing member, in the preferred embodiments steps 940 and 970 have the same impact on remaining members in the group. If the user does not specify to abort the protocol (step 964=NO), the user can then perform some other user-specified operations to determine and remediate the cause of the failure (step 980). For example, the user could send a query message to members of a group. If one member is waiting for something local to occur (such as obtaining a local resource), the main thread for that member can report the status to the user. The user can then take appropriate action, such as killing a job that currently has that resource, so the current job can run to completion.

A passive liveness monitor in accordance with the preferred embodiments is possible due to a main thread that is always responsive (by having no ACK rounds and by not being able to get stuck doing local operations), due to a protocol definition that performs only local work between ACK rounds, and due to the ability to unregister with CLUE, which causes CLUE to generate a membership change that is seen by all the members.

One skilled in the art will appreciate that many variations are possible within the scope of the present invention. Thus, while the invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the art that these and other changes in form and details may be made
5 therein without departing from the spirit and scope of the invention.

We claim:

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2